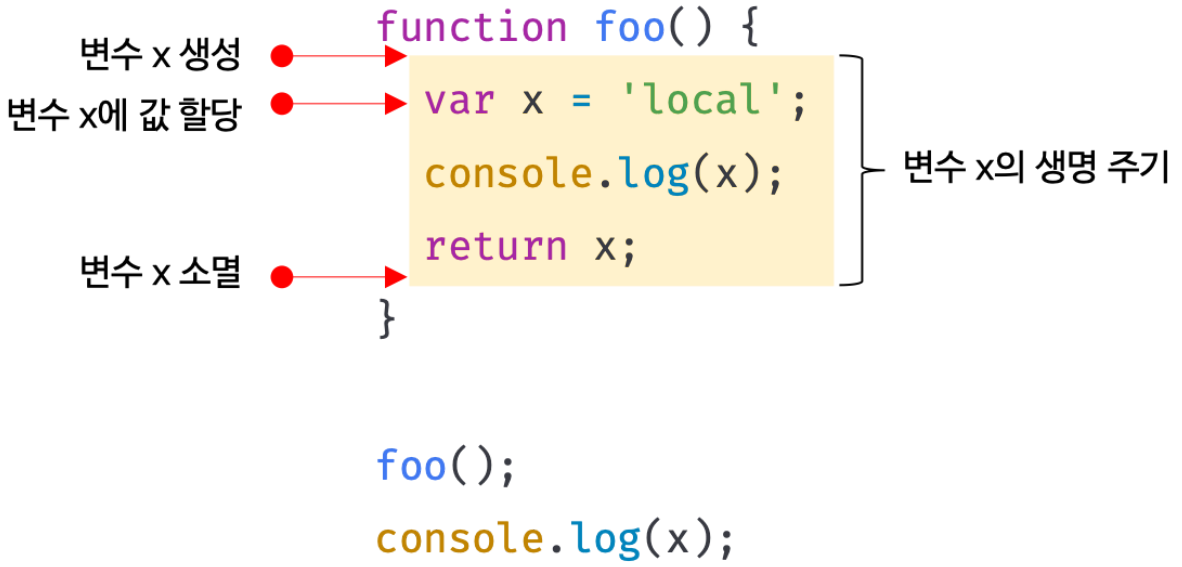
14장 전역 변수의 문제점

14.1 변수의 생명 주기



https://velog.io/@hang_kem_0531/TIL-모던-자바스크립트-Deep-Dive-전역-변수의-문제점

- 지역 변수의 생명 주기는 함수의 생명 주기와 일치한다.

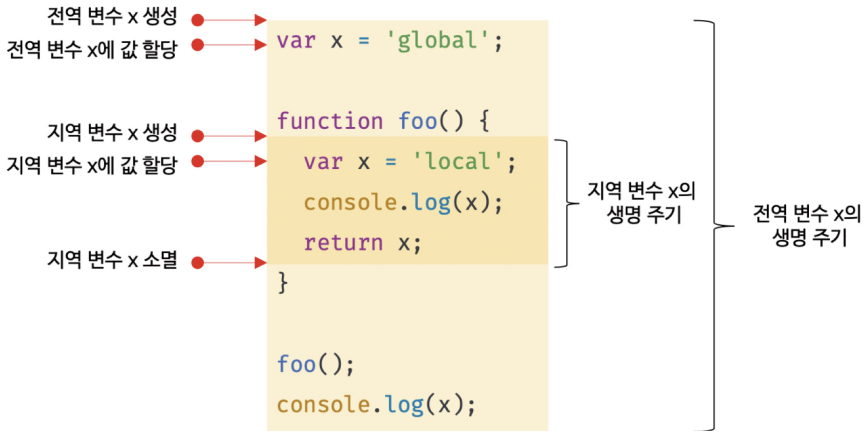
```
var x = 'global';

function foo() {
    console.log(x); // undefined
    var x = 'local';
}

foo()
console.log(x); // global
```

- 호이스팅은 스코프를 단위로 동작한다.
    - 위 코드에서 foo 함수 내부의 console.log(x); 안의 x 값은 전역 변수 x가 아닌 foo 함수 내부에서 선언된 x이며, 호이스팅을 통해 undefined로 우선 선언된 값을 갖는다.

- var 키워드로 선언한 전역 변수의 생명 주기는 전역 객체의 생명 주기와 일치한다.
  - 브라우저 환경에서 전역 객체는 window이므로 브라우저 환경에서 var 키워드로 선언한 전역 변수는 전역 객체 window의 프로퍼티다.
  - 또한 전역 객체 window는 웹페이지를 닫기 전까지 유효하며, 따라서 브라우저 환경에서 var 키워드로 선언한 전역 변수는 웹페이지를 닫을 때까지 유효하다.



https://velog.io/@kozel/모던-자바스크립트-14장-전역-변수의-문제점

> 💡 전역 객체
> 전역 객체는 코드가 실행되기 이전 단계에 자바스크립트 엔진에 의해 어떤 객체보다도 먼저 생성되는 특수한 객체다.

14.2 전역 변수의 문제점

1. 암묵적 결합

- 변수의 유효 범위가 크면 클수록 코드의 가독성은 나빠지고 의도치 않게 상태가 변경될 수 있는 위험성도 높아진다.

2. 긴 생명 주기

- 전역 변수는 생명 주기가 길다. 따라서 메모리 리소스도 오랜 기간 소비한다.
- 전역 변수의 상태를 변경할 수 있는 시간도 길고 기회도 많다.

- 전역 변수는 변수의 중복 선언을 허용하는 var 키워드에 의해 의도치 않은 재할당이 이뤄질 가능성이 있다.

3. 스코프 체인 상에서 종점에 존재

- 변수를 검색할 때 전역 변수가 가장 마지막에 검색되며, 즉 검색 속도가 가장 느리다.

4. 네임스페이스 오염

- 파일이 분리되어 있다 해도 하나의 전역 스코프를 공유한다.
- 이는 다른 파일 내에서 동일한 이름으로 명명된 전역 변수나 함수가 같은 스코프 내에 존재할 경우 위험이 존재한다.

14.3 전역 변수의 사용을 억제하는 방법

- 대체로 지역 변수의 사용
  - 변수의 스코프는 좁을수록 좋다.

1. 즉시 실행 함수

- 모든 코드를 즉시 실행 함수로 감싸면 모든 변수는 즉시 실행 함수의 지역 변수가 된다.

```
(function () {
    var foo = 10; // 즉시 실행 함수의 지역 변수
    // ...
}());

console.log(foo); // ReferenceError: foo is not defined
```

2. 네임스페이스 객체

- 전역에 네임스페이스 역할을 담당할 객체를 생성하고 전역 변수처럼 사용하고 싶은 변수를 프로퍼티에 추가한다.
- 식별자 충돌 방지 효과는 있으나 네임스페이스 객체 자체가 전역 변수에 할당되므로 그다지 유용하지는 않다.

```
var MYAPP = {}; // 전역 네임스페이스 객체

MYAPP.name = 'Lee';

console.log(MYAPP.name); // Lee
```

3. 모듈 패턴

- 클래스를 모방해서 관련이 있는 변수와 함수를 모아 즉시 실행 함수로 감싸 하나의 모듈을 만든다.

```
var Counter = (function () {
    // privare 변수
    var num = 0;

    // 외부로 공개할 데이터나 메서드를 프로퍼티로 추가한 객체를 반환한다.
    return {
        increase() {
            return ++num;
        },
        decrease() {
            return --num;
        }
    };
}());

// private 변수는 외부로 노출되지 않는다.
console.log(Counter.num); // undefined

console.log(Counter.increase()); // 1
console.log(Counter.increase()); // 2
console.log(Counter.decrease()); // 1
console.log(Counter.decrease()); // 0
```

4. ES6 모듈

- ES6 모듈은 파일 자체의 독자적인 모듈 프로퍼티를 제공하기에 더는 전역 변수를 사용할 수 없다.

```
<script type="module" src="lib.mjs"></script>
<script type="module" src="app.mjs"></script>
```